

Ruby master - Feature #15286

Proposal: Add Kernel.#expand(*args)

11/06/2018 11:18 PM - osyo (manga osyo)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		
Description		
This is a suggestion for Hash shorthand.		
<ul style="list-style-type: none">• add support for hash shorthand		
Kernel.#expand(*args) is expand local variable and method to Hash. *args is method or local variable names.		
Example		
<pre>def meth "meth" end a, b, c = 1, 2, 3 # #expand args is local variable or method names # Expand to { a: a, b: b, meth: meth } p expand(:a, :b, :meth) # => {:a=>1, :b=>2, :meth=>"meth"}</pre>		
<pre># Error: `expand': undefined method `hoge' for main:Object (NoMethodError) p expand(:hoge)</pre>		
What can be expanded,		
<ul style="list-style-type: none">• local variable• method		
and, If there are duplicate names, prioritize local variable.		
<pre>def meth "meth" end meth = 42 p expand(:meth) # => {:meth=>42}</pre>		
pull request: https://github.com/ruby/ruby/pull/2006		
Related issues:		
Related to Ruby master - Feature #15236: add support for hash shorthand		Rejected

History

#1 - 11/07/2018 12:33 AM - shevegen (Robert A. Heiler)

I think one possible question in regards to the suggestion here is whether the above method may be useful on its own, even without a shorthand syntax for Hash. (This is really just a question; I personally am not having any strong pro/con opinion.)

I also understand that using a method is different compared to the other two proposals. For example,

```
{ a }
{ a: a }
```

and

```
{x, y}
{x: x, y: y}
```

is different to:

```
meth = 42
p expand(:meth) # => {:meth=>42}
```

So using a method is different to the other two proposals (in the two older issue request, by Ignatius and Shugo Maeda). Perhaps one or more use cases could be described for a new method to be useful even without the hash shorthand notation? I can not think of a good one right now but perhaps others have some ideas.

#2 - 11/07/2018 04:22 AM - osyo (manga osyo)

Thanks for comment, shevegen.
Kernel.#expand can be used as follows.

```
names = [:a, :b, :meth]

# expand(:a, :b, :meth)
expand(*names)
# => {:a=>1, :b=>2, :meth=>"meth"}

# ???
{ *names }
```

#3 - 11/19/2018 02:20 AM - nobu (Nobuyoshi Nakada)

Interesting feature, but I don't think the name Kernel.#expand is acceptable.
Maybe an instance method of Binding?
And I think it should raise a NameError instead of a NoMethodError on invalid names.

#4 - 11/19/2018 02:43 AM - nobu (Nobuyoshi Nakada)

And, what do you expect for keywords, e.g., __FILE__, __LINE__, self, super, and etc?

#5 - 11/21/2018 01:32 PM - osyo (manga osyo)

Thanks nobu :)

Interesting feature, but I don't think the name Kernel.#expand is acceptable.

Yes, I looking for a more good name.
Are there any good names?

Maybe an instance method of Binding?

Yes..., but binding.expand(:a, :b, :c) is long...
Kernel.#expand got the idea from Kernel.#local_variables.

And I think it should raise a NameError instead of a NoMethodError on invalid names.

OK, I try :)

And, what do you expect for keywords, e.g., **FILE**, **LINE**, self, super, and etc?

Yes, I think it is necessary to discuss what to "expand".
This is a simple implementation with methods and local variables

#6 - 11/22/2018 10:31 PM - matz (Yukihiro Matsumoto)

- Related to Feature #15236: add support for hash shorthand added

#7 - 11/22/2018 10:36 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

I am against the idea for some reasons:

- I don't think `expand` is the right name for the behavior
- meta-programming is too much for this half-baked substitute for [#15236](#)

Regarding [#15236](#), we are waiting for the time when our recognition changed to accept the JS behavior. Currently, we (at least me) recognize `{a,b,c}` as a literal for a set, not a shorthand for `{a:a,b:b,c:c}`. I am neutral. I don't want Ruby to follow every JS behavior.

Matz.

#8 - 11/24/2018 08:21 AM - osyo (manga osyo)

OK, Thank you, matz :)